Administrivia
• HW#1 was due about 5 minutes ago… ;)  
  – Feedback?  
• HW#2 is out!  
  – i.e., I had homework too :(  
  – A little more thoughtprovoking and a less handholding

Agenda
• Finish up stacks, look at examples  
• Start queues

Stacks, redux
• Basic operations  
  – Push  
  – Pop  
  – Peek  
  – “LIFO”  
• Extraordinarily simple!

Basic Stack examples
• Reverse a word  
• Conversation  
  – Sentence with parentheses?  
• Delimiter matching: {}()  
  – Conceptually simple to use, less error-prone than array  
• Function/method calls

More complex stack example
• How do computers parse arithmetic expressions?  
• First step: transform expression into postfix notation  
• Second step: evaluate postfix expression using a stack

Postfix
• Also called Reverse Polish Notation (RPN); HP calculators  
• Why?  
  – Parentheses unneeded – no ambiguity  
  – Can process in one pass from left-to-right  
• Fairly straightforward to translate from infix to postfix, but let’s hold off on this

Evaluating a Postfix expression
• Go left-to-right  
  – If operand, push on stack
– If operator, pop two operands, use operator, and push result on stack
– When done, there should be one value on the stack
  – Pop it

9 Converting Infix to Postfix
  • See pages 158-159, although I think my slides make more sense ;)
  • Need to encode operator precedence
  • To process:
    – Operand: write straight to output
    – ( : push on stack
    – ): pop all items until ( encountered, and output them; don’t write the ( 
    – Operator: interesting problem

10 Converting Infix-to-Postfix (II)
  • Operator handling
    – If stack is empty, push
    – Else, pop, determine precedence of new vs. popped
      • If popped is a (, put it back on the stack, and put the new operator on top
      • Else if new has higher precedence, push popped back on, and push new on top of it
      • Else if popped has higher or equal precedence, output it, and repeat this process
    • (PEMDAS for precedence
  • No more?
    – Pop, output repeatedly

11 Queues
  • FIFO, instead of LIFO
  • “Standing in line”: print queue
  • Insert: places at rear of queue
  • Remove: takes from front
  • Peek: looks at front
  • Book’s convention: front is at bottom, near beginning of array
  • Problem: how to represent in array?
    – We can’t stick it at one end or the other, unless we slide all the elements around
    – There’s a better approach

12 Next time…
  • Circular Queues
  • Priority Queues
  • Linked Lists